IN THE CLAIMS:

Please amend the claims as follows:

1. (Original) A computer-implemented method for parallel processing of requests, comprising:

initiating a primary executing entity configured to perform requests and maintain state information specific to the primary executing entity;

initiating a secondary executing entity configured to perform requests and maintain state information specific to the secondary executing entity;

performing, by the primary executing entity, requests; and

performing, by the secondary executing entity, the requests previously performed by the primary executing entity in a time-delayed and step-wise fashion while the primary executing entity continues to execute requests, whereby each executing entity maintains its own respective state information independent of, and temporally displaced from, the other executing entity.

2. (Original) The method of claim 1, further comprising:

making the performance of the primary executing entity visible to a user; and

making the performance of the secondary executing entity transparent to the user.

3. (Original) The method of claim 1, further comprising:

terminating the primary executing entity; and then

performing, by the secondary executing entity, at least a portion of the requests performed by the terminated primary executing entity and not previously performed by the secondary executing entity.

4. (Original) The method of claim 1, wherein the executing entities are threads.

5. (Original) The method of claim 1, further comprising:

producing output by the primary executing entity;

producing output by the secondary executing entity;

displaying the output produced by the primary executing entity; and

discarding, without displaying, the output produced by the secondary executing entity.

6. (Original) The method of claim 1, further comprising, upon encountering an error by the primary executing entity:

terminating the primary executing entity; and

recovering from the error by returning a user to a request currently being handled by the primary executing entity.

7. (Original) The method of claim 1, further comprising, upon encountering an error by the primary executing entity:

terminating the primary executing entity; and

recovering from the error by returning a user to a request handled by the primary executing entity prior to encountering the error.

8. (Original) The method of claim 1, further comprising, upon encountering an error by the primary executing entity:

terminating the primary executing entity; and

recovering from the error by returning a user to a request within a range of requests between a request being handled by the secondary executing entity and a request being handled by the primary executing entity at the time of encountering the error.

9. (Original) The method of claim 1, wherein the requests are time ordered and processed by each performing executing entity according to the time order.

10. (Original) The method of claim 9, further comprising:

terminating the primary executing entity; and then

performing, by the secondary executing entity, the requests performed by the terminated primary executing entity between a last request handled by the terminated primary executing entity and a last request handled by the secondary executing entity.

11. (Original) A computer-implemented method for parallel processing of requests, comprising:

receiving user requests from a user;

placing the user requests on a queue in a time-ordered manner;

performing, by a primary executing entity, each current user request upon being placed on the queue; and

performing, by a secondary executing entity, at least a portion of the user requests on the queue step-wise with the primary executing entity and N-requests behind the primary executing entity; wherein each of the executing entities maintain their own respective state information.

12. (Original) The method of claim 11, further comprising:

encountering an error by the primary executing entity;

terminating the primary executing entity; and

recovering from the error by returning a user to a request being handled by the primary executing entity when being terminated.

13. (Cancelled)

14. (Original) The method of claim 11, wherein the executing entities are threads.

15. (Original) The method of claim 11, wherein executing entities are executable code elements of an application.

16. (Original) A computer-implemented method for parallel processing of requests, comprising:

receiving user requests from a user;

placing the user requests on a queue in a time-ordered manner;

performing, by a primary executing entity, each current user request upon being placed on the queue, wherein the performance of requests on the queue by the primary executing entity are visible to the user; and

performing, by a plurality of secondary executing entities, at least a portion of the user requests on the queue step-wise with the primary executing entity and in a manner transparent to the user; wherein each of the secondary executing entities is displaced from the primary executing entity and from one another by a number of user requests; and wherein each of the executing entities maintain their own respective state information.

17. (Original) The method of claim 16, wherein the executing entities are threads.

18. (Original) The method of claim 16, wherein executing entities are executable code elements of an application.

19. (Original) The method of claim 16, further comprising making one of the secondary executing entities visible to the user upon encountering an error by the primary executing entities.

20. (Original) The method of claim 16, further comprising upon encountering an error by the primary executing entity:

terminating the primary executing entity; and

recovering from the error by returning a user to a request currently being handled by the primary executing entity immediately prior to being terminated.

21. (Previously Presented) A computer readable storage medium containing a program which, when executed, implements an operation for parallel processing of requests, the operation comprising:

initiating a primary executing entity configured to perform requests and maintain state information specific to the primary executing entity;

initiating a secondary executing entity configured to:

perform requests previously performed by the primary executing entity in a time-delayed and step-wise fashion while the primary executing entity continues to execute requests; and

maintain state information specific to the secondary executing entity, whereby each executing entity maintains its own respective state information independent of, and temporally displaced from, the other executing entity.

22. (Previously Presented) The computer readable storage medium of claim 21, the operation further comprising:

making the performance of the primary executing entity visible to a user; and

making the performance of the secondary executing entity transparent to the user.

23. (Previously Presented) The computer readable storage medium of claim 21, the operation further comprising:

terminating the primary executing entity; and then

performing, by the secondary executing entity, requests previously performed by the terminated primary executing entity and not previously performed by the secondary executing entity.

24. (Previously Presented) The computer readable storage medium of claim 21, wherein the executing entities are threads.

25. (Previously Presented) The computer readable storage medium of claim 21, further comprising:

producing output by the primary executing entity;

producing output by the secondary executing entity;

displaying the output produced by the primary executing entity; and

discarding, without displaying, the output produced by the secondary executing entity.

26. (Previously Presented) The computer readable storage medium of claim 21, further comprising, upon encountering an error by the primary executing entity:

terminating the primary executing entity; and

recovering from the error by returning a user to a request currently being handled by the primary executing entity immediately prior to being terminated.

27. (Previously Presented) The computer readable storage medium of claim 21, wherein the requests are time ordered and processed by each performing executing entity according to the time order.

28. (Previously Presented) The computer readable storage medium of claim 27, further comprising:

terminating the primary executing entity; and then

performing, by the secondary executing entity, the requests performed by the terminated primary executing entity between a last request handled by the terminated primary executing entity and a last request handled by the secondary executing entity.

29. (Previously Presented) A computer readable storage medium containing a program which, when executed, implements an operation for parallel processing of requests, the operation comprising:

receiving user requests from a user;

placing the user requests on a queue in a time-ordered manner;

performing, by a primary executing entity, each current user request upon being placed on the queue; and

performing, by a secondary executing entity, at least a portion of the user requests on the queue step-wise with the primary executing entity and N-requests behind the primary executing entity; wherein each of the executing entities maintain their own respective state information.

30. (Previously Presented) The computer readable storage medium of claim 29, further comprising, upon encountering an error by the primary executing entity:

terminating the primary executing entity; and

recovering from the error by returning a user to a request currently being handled by the primary executing entity immediately prior to being terminated.

31. (Cancelled)

32. (Previously Presented) The computer readable storage medium of claim 29, wherein the executing entities are threads.

33. (Previously Presented) The computer readable storage medium of claim 29, wherein executing entities are executable code elements of an application.

34. (Previously Presented) A computer readable storage medium containing a program which, when executed, implements an operation for parallel processing of requests, the operation comprising:

receiving user requests from a user;

placing the user requests on a queue in a time-ordered manner;

performing, by a primary executing entity, each current user request upon being placed on the queue, wherein the performance of requests on the queue by the primary executing entity are visible to the user; and

performing, by a plurality of secondary executing entities, at least a portion of the user requests on the queue step-wise with the primary executing entity and in a manner transparent to the user; wherein each of the secondary executing entities is displaced from the primary executing entity and from one another by a number of user requests; and wherein each of the executing entities maintain their own respective state information.

35. (Previously Presented) The computer readable storage medium of claim 34, wherein the executing entities are threads.

36. (Previously Presented) The computer readable storage medium of claim 34, wherein executing entities are executable code elements of an application.

37. (Previously Presented) The computer readable storage medium of claim 34, further comprising making one of the secondary executing entities visible to the user upon encountering an error by the primary executing entities.

38. (Previously Presented) The computer readable storage medium of claim 34, further comprising:

encountering an error by the primary executing entity;

terminating the primary executing entity; and

recovering from the error by returning a user to a request currently being handled by the primary executing entity immediately prior to being terminated.

39. (Original) A system for configured for parallel processing of requests, comprising:

a request queue on which requests are placed in a time-ordered manner;

a primary executing entity configured to process requests from the request queue;

at least one secondary executing entity configured to process requests from the request queue in step-wise and a time-delayed with respect to the primary executing entity, whereby the at least one secondary executing entity performs request processing N requests behind the primary executing entity; and wherein each of the executing entities maintain their own respective state information; and

a request manager configured to:

make the primary executing entity visible to a user placing the requests on the request queue and make the secondary executing entity transparent to the user, until encountering an error; and

upon encountering the error, terminate the primary executing entity and make the secondary executing entity visible to the user, whereby the user is given the appearance of being returned to a prior request in the execution path the primary executing entity.

40. (Original) The system of claim 39, wherein, upon encountering the error, the request manager is further configured to permit user-selection of replay processing wherein all requests from a request currently being handled by the secondary executing entity and a request currently being handled by the primary executing entity when the error was encountered are preserved and re-executed.

41. (Original) The system of claim 39, wherein, upon encountering the error, the request manager is further configured to permit user-selection of retry processing wherein all requests from a request currently being handled by the secondary executing entity and a request being handled by the primary executing entity when the error was encountered are discarded and the user is given the opportunity to submit new requests.

42. (Original) The system of claim 39, wherein the at least one secondary executing entity comprises a plurality of secondary executing entities each displaced from the primary executing entity and from one another by a number of user requests.